Multipartite RRTs for Rapid Replanning in Dynamic Environments

By: Mat Zucker, James Kuffner, Michael Branicky

ICRA 2007

Presented by: Brad Woosley

CSCI 9420
Introduction

- Variant of Rapidly-exploring Random Tree (RRT)
- Opportunistic strategy to reuse previous query information
Background

- Dynamic planning
  - Robot moves between planning iterations
  - Goal can move
  - Obstacles can move
- Can re-run entire planning algorithm anytime something changes
  - Doesn't take into account that changes are normally small
Background

- ERRT (Bruce, 2002)
  - Each point along path from start to goal stored
    - Waypoint cache
  - If change detected, discard path but sample points from waypoint cache with probability $p_{\text{waypoint}}$
Background

- **DRRT (Ferguson, 2006)**
  - Check previous RRT for collisions
  - Keep and try to reuse branches of original tree that are still in $C_{\text{free}}$ and connected to root

- **Reconfigurable Random Forests (RRF) (Hirano, 2005)**
  - Grow multiple disconnected RRTs back from goal to start point
Multipartite RRT (MP-RRT) Algorithm

- $G =$ set of goal locations
- Combines strengths of ERRT and DRRT
- If no tree built, or start/goal states changed substantially
  - Build new RRT from scratch
- Maintains a forest $F$ of subtrees not connected to root of $T$
- Edges of $F$ and $T$ not valid are removed
- Any new disconnected subtrees are added to $F$
If start is no longer root of tree, tries to connect tree to new start point
If fails, adds old tree to F
  - T contains single point, the new start location
If goal isn't in T, run RRT, but bias sampling towards the root of a tree in F
**Procedure** MPRRTSEARCH($q_{init}$)
Performs the MP-RRT algorithm.

**Data:** $T$: the previous search tree, if any
$F$: the previous forest of disconnected subtrees
$q_{init}$: the initial state

**Result:** a boolean value indicating plan success

if EMPTY($T$) then
  $q_{root} = q_{init}$;
  INSERT($q_{root}, T$);
else
  PRUNEANDPREPEND($T, F, q_{init}$);
  if TREEHASGOAL($T$) then
    return true
  end
end

while search time/space remaining do
  $q_{new} = SELECTSAMPLE(F)$;
  $q_{nearest} = NEARESTNEIGHBOR(q_{new}, T)$;
  if $q_{new} \in F$ then
    $b_{connect} = CONNECT(q_{nearest}, q_{new})$;
    if $b_{connect}$ and TREEHASGOAL($T$) then
      return true
    end
  else
    $b_{extend} = EXTEND(q_{nearest}, q_{new})$;
    if $b_{extend}$ and ISGOAL($q_{new}$) then
      return true
    end
  end
end

return false ;
Procedure `PRUNEANDPREPEND`\((T, F, q_{init})\)
Used at start of MP-RRT search query.

**Data:**
- \(T\): the original search tree
- \(F\): the forest of disconnected subtrees
- \(q_{init}\): the new initial state

**Result:** The search tree \(T\) is valid and the forest \(F\) contains any surviving subtrees of the original search tree.

```
for each node \(q \in T, F\) do
    if not NODEVALID\((q)\) then
        KILLNODE\((q)\);
    else if not ACTIONVALID\((q)\) then
        SPLITEDGE\((q)\);
    end
end

if not EMPTY\((T)\) and \(q_{root} \neq q_{init}\) then
    if not REROOT\((T, q_{init})\) then
        place old \(T\) in \(F\);
        initialize \(T\) to have \(q_{root} = q_{init}\);
    end
end
```
Procedure SELECTSAMPLE(F)
Used to generate samples for MP-RRT.

Data: \( F \): the forest of disconnected subtrees

Result: \( q_{new} \): the selected sample state

\[
p = \text{RANDOM}(0, 1);
\]

\[
\text{if } p < p_{goal} \text{ then }
\]
\[
q_{new} = q_{goal} \in G;
\]

\[
\text{else if } p < (p_{goal} + p_{forest}) \text{ and not EMPTY}(F) \text{ then }
\]
\[
q_{new} = q \in \text{SUBTREEROOTS}(F);
\]

\[
\text{else}
\]
\[
q_{new} = \text{RANDOMSTATE}();
\]

\[
\text{return } q_{new} ;
\]
MP-RRT

- Can be easily transformed into previous algorithms
- Delete tree instead of adding to F
  - DRRT
- Invalidate all edges of T at start of planning
  - ERRT
MP-RRT

- Selection of parameter $p_{\text{forest}}$
- Tradeoff between collision checking and integrating subtrees
- Tuned to result in a ratio of failed to successful connections of 2:1 or 3:1
  - Significant performance gains of DRRT
Heuristics

- First sample a goal state
- Then sample the root of a tree in F that contains a goal state
- Grow long branches with intermediate waypoints
- NearestNeighbor function return closest neighbor that an extension has not previously failed on
Heuristics

• Upper bound on size of $F$
  – Discard if haven't been connected to $T$ after some number of planning iterations
Experiments

- 2D, 3D, and 4D experiments
- Simulation
- Points sampled uniformly from a rectangle
2D and 3D

- Limited sensing abilities
- 2D plane with various sized circular obstacles
- 2D: robot is disc shaped translating robot
- 3D: rectangular robot
• 4D global sensing model, obstacle's position, velocity known, but moved unpredictably

• Time sampled from $[t_0, 2t_{\text{min}}(x,y,\theta)]$
  - $t_0$ initial time of planner
  - $t_{\text{min}}(x,y,\theta)$ gives min time to reach random sampled point $(x,y,\theta)$ neglecting obstacles

• Prune if time for configuration if F or T is before the time of the start configuration
Experiments

- Limited velocity in $x, y, \theta$, but unlimited acceleration
- Angle sampled in range $[0, 2\pi)$
  - (3D and 4D)
- Definition of $G$:
  - states around $(x_{\text{goal}}, y_{\text{goal}})$ with fixed radius (2D)
  - Cylinder swept along $\theta$ (3D & 4D)
Trial procedure

- Random distribution of obstacles
- Start and goal locations randomly on opposite sides of environment
- Plan path from start to goal
  - < 100 samples per planning iteration
  - < 5000 nodes total in final RRT
  - Helps since robot at start doesn't have complete info about environment
Trial procedure

- Step robot along path in RRT which ends closest to goal region
- Repeat plan/step until:
  - Goal reached
  - Obstacle collision (4D only)
  - Tree is full without a path found
Algorithms tested

- **MP-RRT**
  - Forest bias = 0.1
- **DRRT**
- **ERRT**
  - Waypoint bias = 0.5
- **Naive iterated RRT**
- **For all algorithms**
  - Goal bias at 0.05
- **For a set of trials, initial location of goal, start, and obstacles were the same for all algorithms**
Smoothing

- Also ran with and without a greedy smoothing algorithm
- Shorten path through skipping nodes
- Produces paths close to obstacles

**Procedure** SMOOTHPATH($P$)

Used to smooth paths to eliminate unnecessary detours

**Data:** $P$: path ($q_0, \ldots, q_N$) computed by the RRT

**Result:** $P'$: shortened path containing elements of $P$

```plaintext
for each node $q_i \in q_N, \ldots, q_1$ do
  if CONNECT($q_0, q_i$) then
    break;
  end
end

$P' = (q_0, q_i, q_{i+1}, \ldots, q_N)$;

return $P'$;
```
Results

• 100 randomized trials of 2D and 3D test
• 1000 randomized trials of 4D
• Data recorded:
  – If robot reached goal
  – Number of samples generated
  – Number of collision checks on edges in the RRT
  – Number of planning iterations
Results 2D / 3D

- MP-RRT as good or better for successful trials
- Smoothing didn't alter DRRT and MP-RRT results significantly
- ERRT and naïve RRT benefited from smoothing
  - Still didn't do better than DRRT or MP-RRT
# Results

<table>
<thead>
<tr>
<th>Setup</th>
<th>Algorithm</th>
<th>Successful</th>
<th>Samples</th>
<th>Edge CC</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>2D</td>
<td>Iter. RRT</td>
<td>92 / 100</td>
<td>202,134</td>
<td>283,242</td>
<td>20.77s</td>
</tr>
<tr>
<td></td>
<td>ERRT</td>
<td>96 / 100</td>
<td>113,548</td>
<td>163,682</td>
<td>12.03s</td>
</tr>
<tr>
<td></td>
<td>DRRT</td>
<td>99 / 100</td>
<td>31,821</td>
<td>120,107</td>
<td>5.90s</td>
</tr>
<tr>
<td></td>
<td>MP-RRT</td>
<td>99 / 100</td>
<td>25,346</td>
<td>100,278</td>
<td>4.81s</td>
</tr>
<tr>
<td>3D</td>
<td>Iter. RRT</td>
<td>62 / 100</td>
<td>546,363</td>
<td>667,554</td>
<td>69.52s</td>
</tr>
<tr>
<td></td>
<td>ERRT</td>
<td>71 / 100</td>
<td>437,689</td>
<td>527,165</td>
<td>53.65s</td>
</tr>
<tr>
<td></td>
<td>DRRT</td>
<td>88 / 100</td>
<td>223,207</td>
<td>400,639</td>
<td>37.48s</td>
</tr>
<tr>
<td></td>
<td>MP-RRT</td>
<td>82 / 100</td>
<td>238,026</td>
<td>416,548</td>
<td>39.23s</td>
</tr>
<tr>
<td>4D</td>
<td>Iter. RRT</td>
<td>706 / 1000</td>
<td>1,668,316</td>
<td>2,319,569</td>
<td>298.66s</td>
</tr>
<tr>
<td></td>
<td>ERRT</td>
<td>712 / 1000</td>
<td>1,487,460</td>
<td>2,041,470</td>
<td>260.25s</td>
</tr>
<tr>
<td></td>
<td>DRRT</td>
<td>783 / 1000</td>
<td>1,705,502</td>
<td>3,079,187</td>
<td>366.54s</td>
</tr>
<tr>
<td></td>
<td>MP-RRT</td>
<td>825 / 1000</td>
<td>1,401,247</td>
<td>2,815,880</td>
<td>326.72s</td>
</tr>
<tr>
<td>4D*</td>
<td>Iter. RRT</td>
<td>35 / 1000</td>
<td>2,115,648</td>
<td>3,226,302</td>
<td>419.15s</td>
</tr>
<tr>
<td></td>
<td>ERRT</td>
<td>38 / 1000</td>
<td>1,936,447</td>
<td>2,953,861</td>
<td>380.23s</td>
</tr>
<tr>
<td></td>
<td>DRRT</td>
<td>180 / 1000</td>
<td>2,132,522</td>
<td>3,940,813</td>
<td>478.56s</td>
</tr>
<tr>
<td></td>
<td>MP-RRT</td>
<td>509 / 1000</td>
<td>1,680,962</td>
<td>4,101,404</td>
<td>480.65s</td>
</tr>
</tbody>
</table>

*The second set of 4D results use the greedy smoothing technique described in section IV-A.*
Results 4D

- Modestly better than DRRT and ERRT if no smoothing is used
- Larger gap if smoothing is used
- Constructs more robust plans to obstacle movement, even when close to obstacles
Fig. 4. Sorted performance graphs for ERRT (black dash/dot line), DRRRT (blue dashed line), and MP-RRT (red line) on the 4D experimental setup. Only the successful trials from a 1000-trial run are shown. Top row: statistics for trials with “greedy” smoothing turned off. Bottom row: statistics for trials with smoothing turned on. Left: total number of samples generated during an experimental trial. Center: number of edge validation steps performed during a trial. Right: total planning duration in wall clock time.
Results 4D

- MP-RRT requires fewer iterations than DRRT or ERRT for difficult environments

Fig. 5. Sorted performance graph for number of planning iterations per successful trial in 4D setup (both with and without smoothing heuristic). Black dash/dot line is DRRT, blue dashed line is ERRT, solid red line is MP-RRT.
Conclusion

- MP-RRT is well suited to dynamic planning
- Performs better than previous approaches
  - Adapts to unknown obstacles and moving obstacles
Questions?